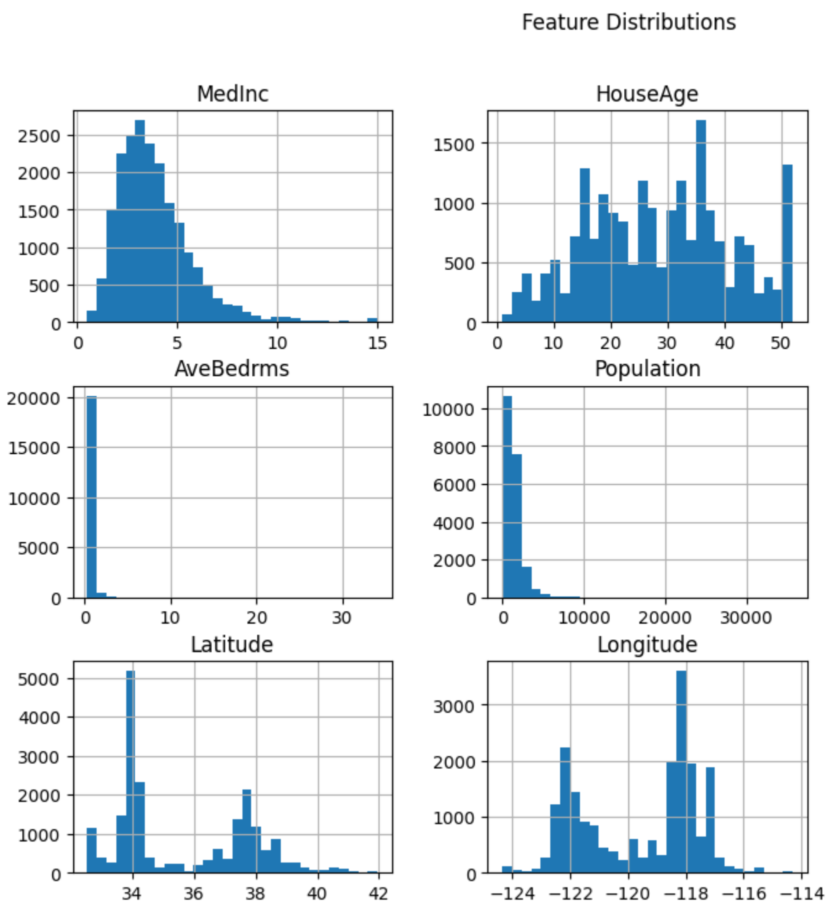
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import LearningRateScheduler
from sklearn.metrics import mean_squared_error, r2_score

# Step 1: Load the California Housing Dataset
data = fetch_california_housing()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target

# Step 2: Exploratory Data Analysis (EDA)

# Visualizing the feature distribution
plt.figure(figsize=(10, 6))
df.drop('target', axis=1).hist(bins=30, figsize=(12, 8))
plt.suptitle('Feature Distributions')
plt.show()

# Correlation heatmap to identify the relationships between features
plt.figure(figsize=(10, 6))
corr = df.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()

# Scatter plot matrix to visualize relationships between features
sns.pairplot(df, vars=df.columns[:-1], height=2.5)
plt.suptitle('Feature Pairwise Scatter Plots', y=1.02)
plt.show()

# Plot target vs. each feature to analyze their impact on the target
fig, axes = plt.subplots(3, 3, figsize=(15, 15))
axes = axes.ravel()
for i, feature in enumerate(df.columns[:-1]):
    sns.scatterplot(x=df[feature], y=df['target'], ax=axes[i])
    axes[i].set_title(f'{feature} vs. Target')
plt.tight_layout()
plt.show()

# Step 3: Preprocess Data (Train-Test Split & Scaling)
X = df.drop('target', axis=1).values
y = df['target'].values
```

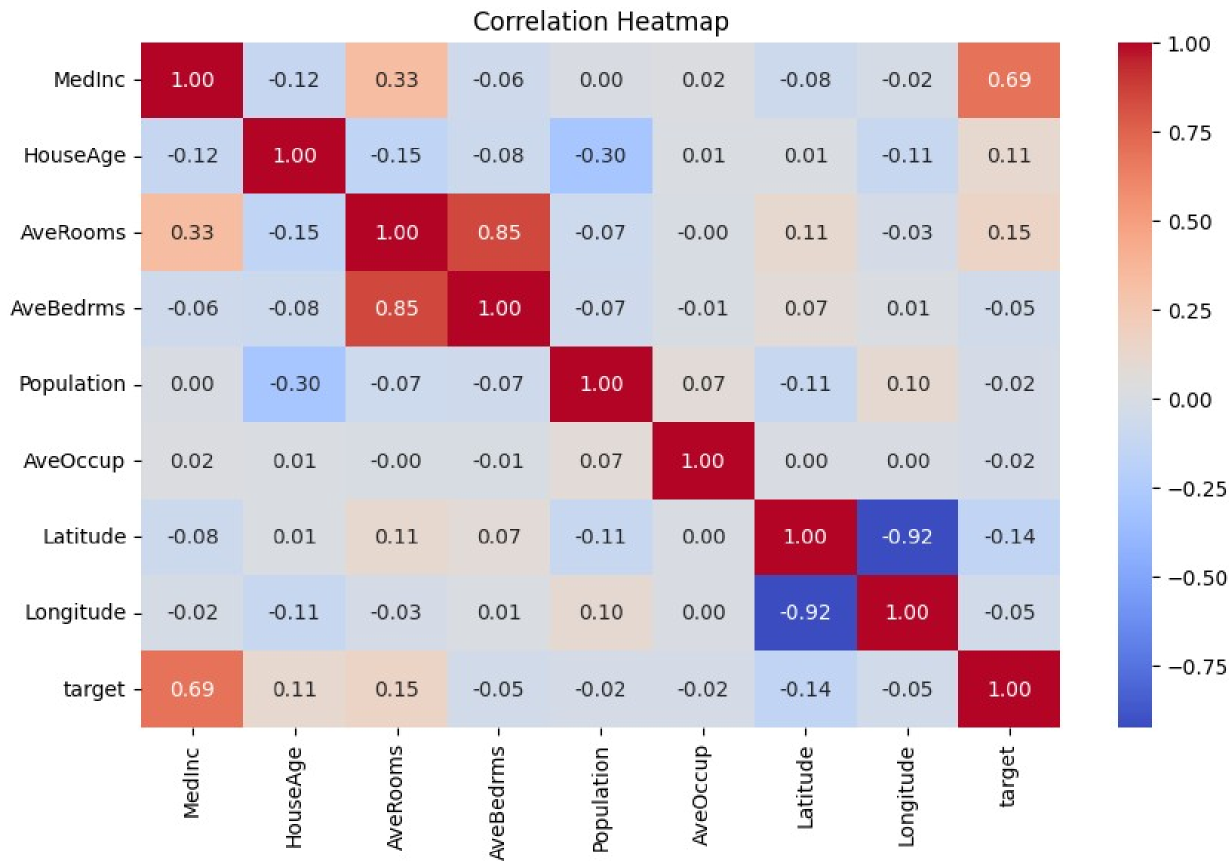
```python
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Scaling the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Step 4: Define Model Architecture (Advanced Deep Learning)
model = Sequential([
    Dense(128, input_dim=X_train.shape[1], activation='relu'),
    BatchNormalization(),
    Dropout(0.5),
    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.5),
    Dense(32, activation='relu'),
    Dense(1)
])

# Compiling the model with Adam optimizer
model.compile(optimizer=Adam(learning_rate=0.001), loss='mse',
metrics=['mae'])

# Learning Rate Scheduler
def scheduler(epoch, lr):
    if epoch < 10:
        return lr
    else:
        return lr * 0.7

lr_scheduler = LearningRateScheduler(scheduler)

# Step 5: Train the Model
history = model.fit(X_train, y_train, epochs=30, batch_size=32,
validation_data=(X_test, y_test), callbacks=[lr_scheduler])

# Step 6: Model Evaluation
y_pred = model.predict(X_test)

# Calculate RMSE and R2 score
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

# Print RMSE and R2 score
print(f"Root Mean Squared Error: {rmse:.4f}")
print(f"R2 Score: {r2:.4f}")

# Step 7: Visualizations
```

```
# 1. Plotting the Training and Validation Loss
plt.figure(figsize=(10, 6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

# 2. Plotting the Training and Validation MAE
plt.figure(figsize=(10, 6))
plt.plot(history.history['mae'], label='Training MAE')
plt.plot(history.history['val_mae'], label='Validation MAE')
plt.title('Training and Validation MAE Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Mean Absolute Error')
plt.legend()
plt.show()

# 3. Learning Rate Decay Plot
lr_rates = [scheduler(epoch, 0.001) for epoch in range(30)]
plt.figure(figsize=(10, 6))
plt.plot(lr_rates, label='Learning Rate Over Epochs')
plt.title('Learning Rate Decay Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Learning Rate')
plt.legend()
plt.show()

# 4. Predicted vs Actual Plot
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
color='red', linewidth=2)
plt.title('Predicted vs Actual Values')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.show()

# ... (previous code) ...

# 5. Residuals Plot (Predictions - Actual Values)
residuals = y_test - y_pred.flatten() # Flatten y_pred to shape
(4128,)
plt.figure(figsize=(10, 6))
plt.scatter(y_pred, residuals)
plt.axhline(0, color='red', linestyle='--')
plt.title('Residuals Plot')
plt.xlabel('Predicted Values')
```

```python
plt.ylabel('Residuals')
plt.show()

# ... (rest of the code) ...

# 6. Feature Importance Plot (using Random Forest for feature
importance estimation)
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

# Plotting feature importances
plt.figure(figsize=(10, 6))
plt.barh(df.columns[:-1], rf.feature_importances_)
plt.title('Feature Importance')
plt.xlabel('Importance')
plt.ylabel('Features')
plt.show()

# 7. Actual vs Predicted Distribution
plt.figure(figsize=(10, 6))
sns.histplot(y_test, color='blue', label='Actual', kde=True)
sns.histplot(y_pred, color='red', label='Predicted', kde=True)
plt.title('Actual vs Predicted Distribution')
plt.xlabel('House Price')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

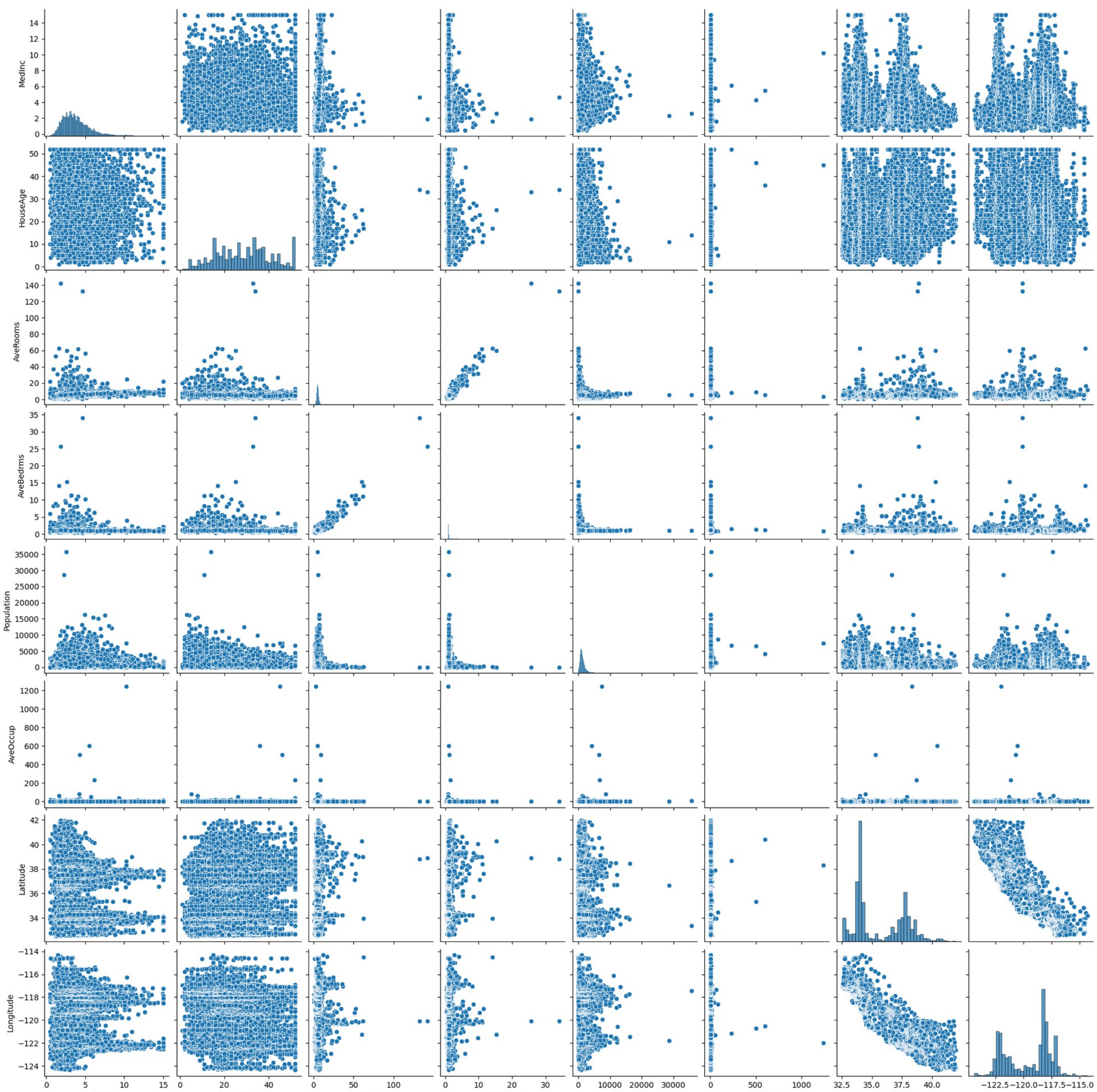
```
<Figure size 1000x600 with 0 Axes>
```

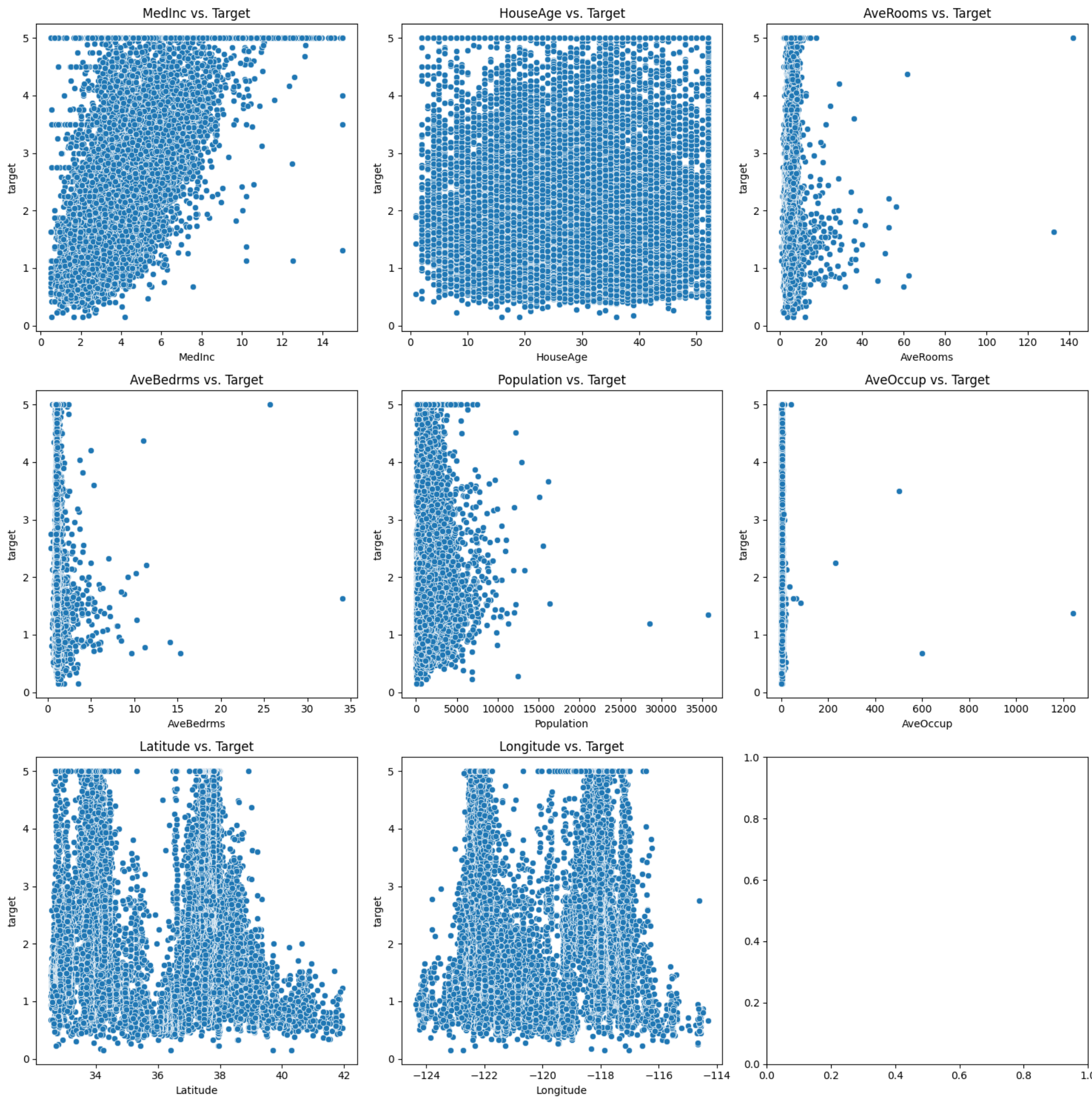

Feature Distributions
MedInc
HouseAge
AveRooms
AveBedrms
Population
AveOccup
Latitude
Longitude



Correlation Heatmap
MedInc
HouseAge
AveRooms
AveBedrms
Population
AveOccup
Latitude
Longitude
target
1.00 -0.12 0.33 -0.06 0.00 0.02 -0.08 -0.02 0.69
-0.12 1.00 -0.15 -0.08 -0.30 0.01 0.01 -0.11 0.11
0.33 -0.15 1.00 0.85 -0.07 -0.00 0.11 -0.03 0.15
-0.06 -0.08 0.85 1.00 -0.07 -0.01 0.07 0.01 -0.05
0.00 -0.30 -0.07 -0.07 1.00 0.07 -0.11 0.10 -0.02
0.02 0.01 -0.00 -0.01 0.07 1.00 0.00 0.00 -0.02
-0.08 0.01 0.11 0.07 -0.11 0.00 1.00 -0.92 -0.14
-0.02 -0.11 -0.03 0.01 0.10 0.00 -0.92 1.00 -0.05
0.69 0.11 0.15 -0.05 -0.02 -0.02 -0.14 -0.05 1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Feature Pairwise Scatter Plots

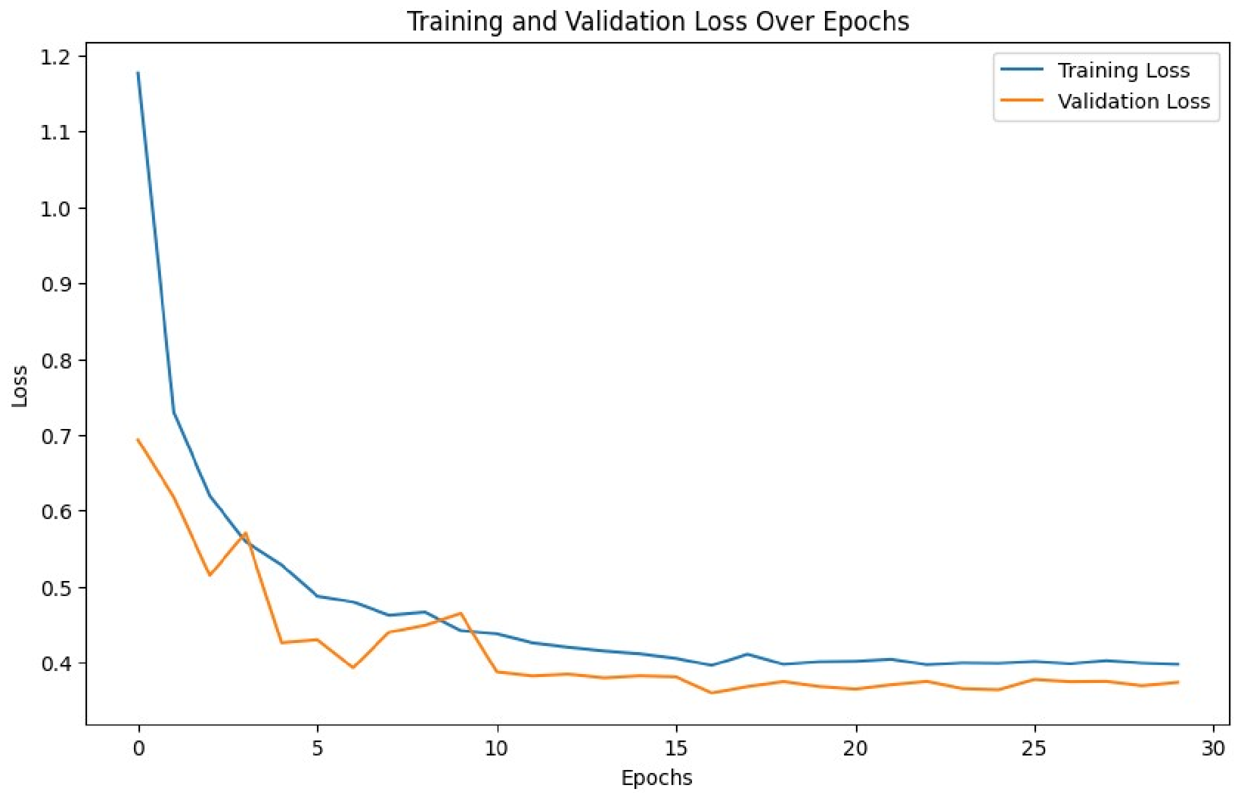


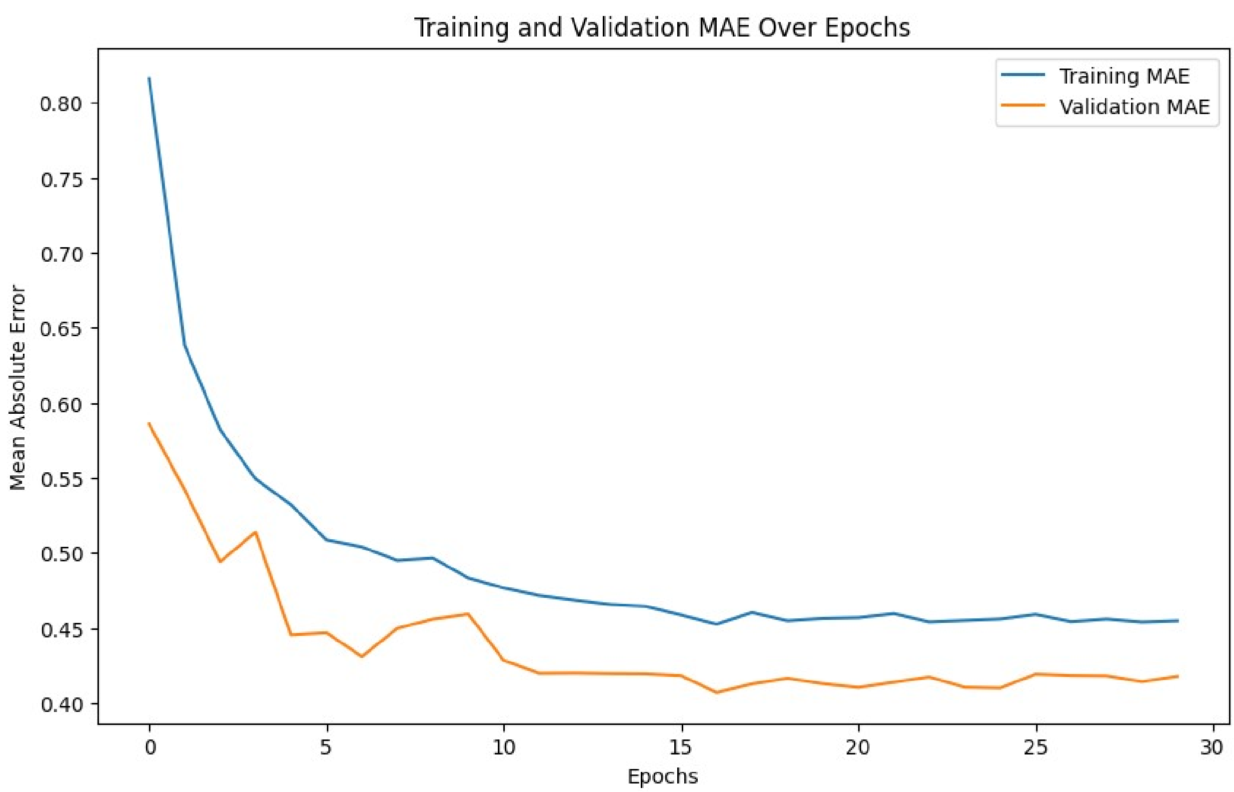


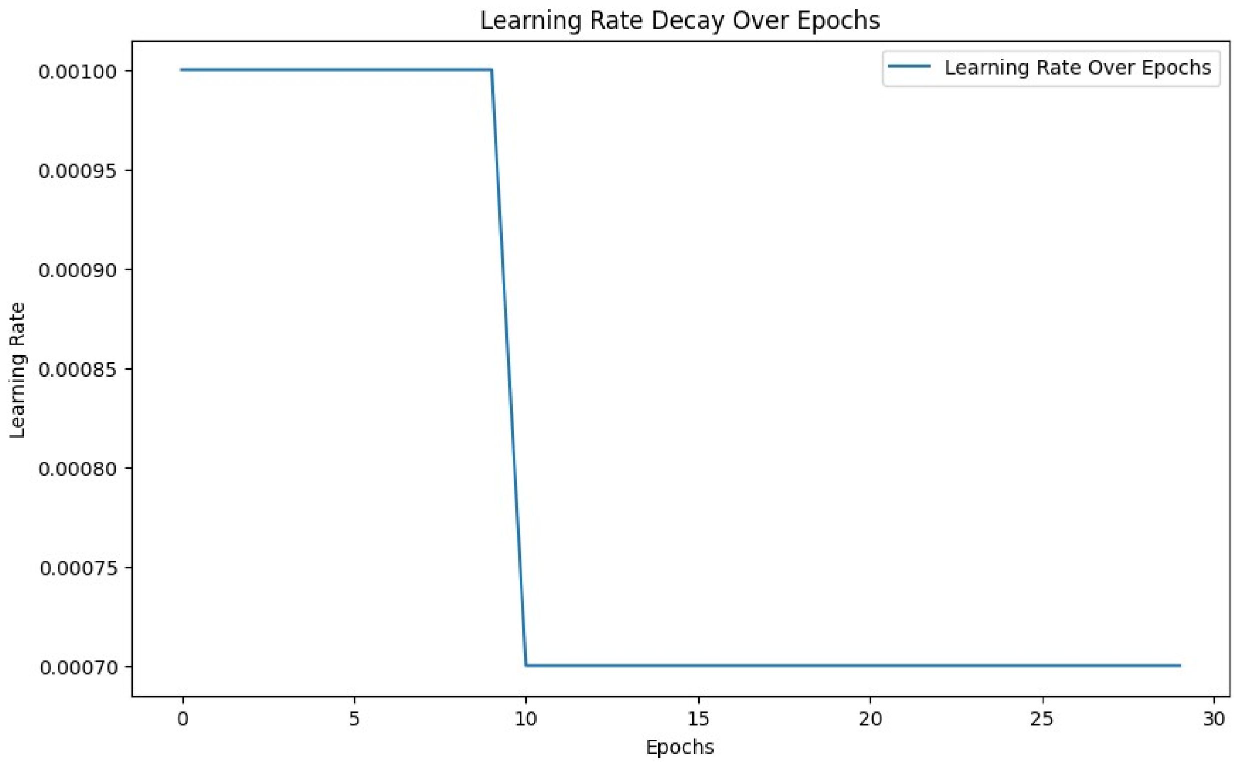

```
Epoch 1/30

/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/
dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an
`Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

516/516 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - loss: 1.6134 - mae: 0.9545
- val_loss: 0.6935 - val_mae: 0.5859 - learning_rate: 0.0010
Epoch 2/30
```

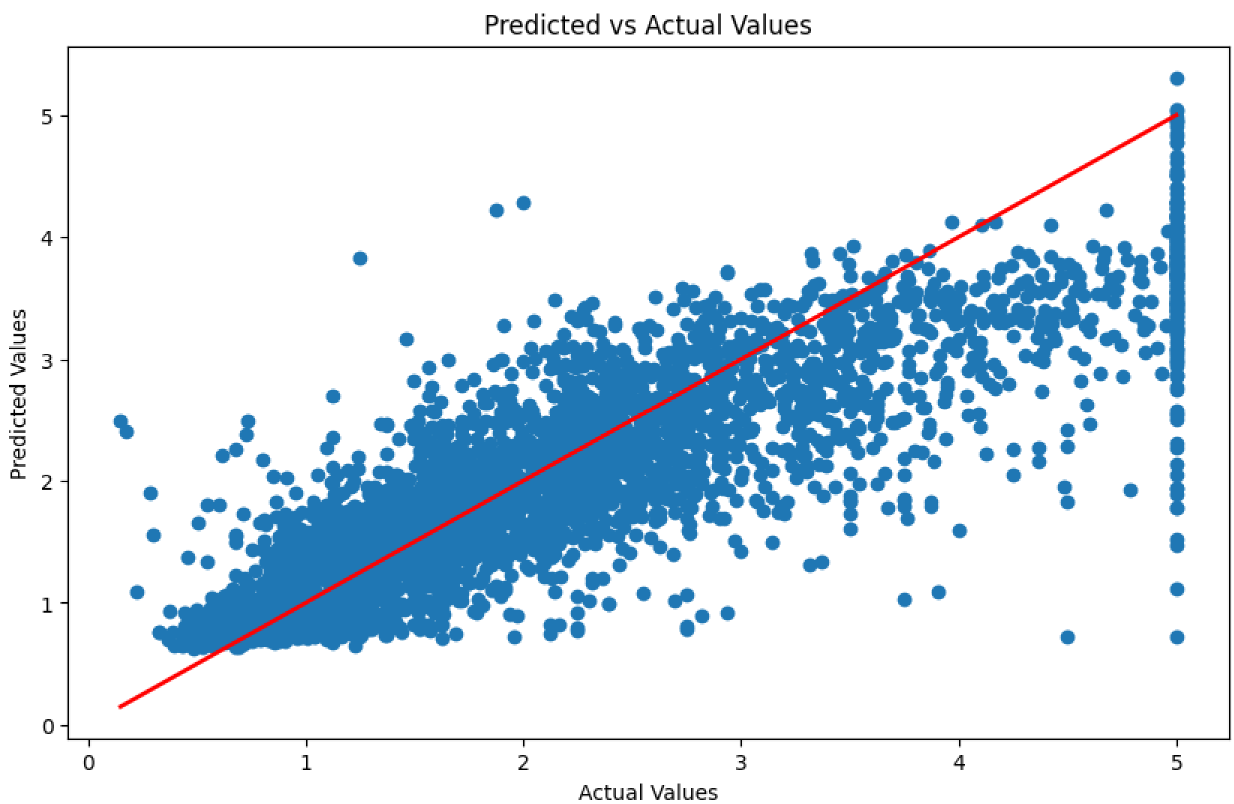
```
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.7454 - mae: 0.6526
- val_loss: 0.6177 - val_mae: 0.5421 - learning_rate: 0.0010
Epoch 3/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.6273 - mae: 0.5914
- val_loss: 0.5148 - val_mae: 0.4942 - learning_rate: 0.0010
Epoch 4/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.5616 - mae: 0.5514
- val_loss: 0.5709 - val_mae: 0.5138 - learning_rate: 0.0010
Epoch 5/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 3s 5ms/step - loss: 0.5278 - mae: 0.5334
- val_loss: 0.4259 - val_mae: 0.4454 - learning_rate: 0.0010
Epoch 6/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 3s 5ms/step - loss: 0.4894 - mae: 0.5114
- val_loss: 0.4299 - val_mae: 0.4469 - learning_rate: 0.0010
Epoch 7/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.4800 - mae: 0.5081
- val_loss: 0.3930 - val_mae: 0.4310 - learning_rate: 0.0010
Epoch 8/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.4722 - mae: 0.4999
- val_loss: 0.4399 - val_mae: 0.4498 - learning_rate: 0.0010
Epoch 9/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.4682 - mae: 0.4992
- val_loss: 0.4491 - val_mae: 0.4560 - learning_rate: 0.0010
Epoch 10/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.4271 - mae: 0.4771
- val_loss: 0.4647 - val_mae: 0.4593 - learning_rate: 0.0010
Epoch 11/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.4377 - mae: 0.4781
- val_loss: 0.3876 - val_mae: 0.4284 - learning_rate: 7.0000e-04
Epoch 12/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.4190 - mae: 0.4715
- val_loss: 0.3823 - val_mae: 0.4199 - learning_rate: 4.9000e-04
Epoch 13/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.4171 - mae: 0.4647
- val_loss: 0.3846 - val_mae: 0.4201 - learning_rate: 3.4300e-04
Epoch 14/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.4112 - mae: 0.4645
- val_loss: 0.3797 - val_mae: 0.4197 - learning_rate: 2.4010e-04
Epoch 15/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.4224 - mae: 0.4710
- val_loss: 0.3824 - val_mae: 0.4196 - learning_rate: 1.6807e-04
Epoch 16/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.4006 - mae: 0.4551
- val_loss: 0.3811 - val_mae: 0.4183 - learning_rate: 1.1765e-04
Epoch 17/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 3s 5ms/step - loss: 0.3936 - mae: 0.4495
- val_loss: 0.3600 - val_mae: 0.4071 - learning_rate: 8.2354e-05
Epoch 18/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.4125 - mae: 0.4593
```

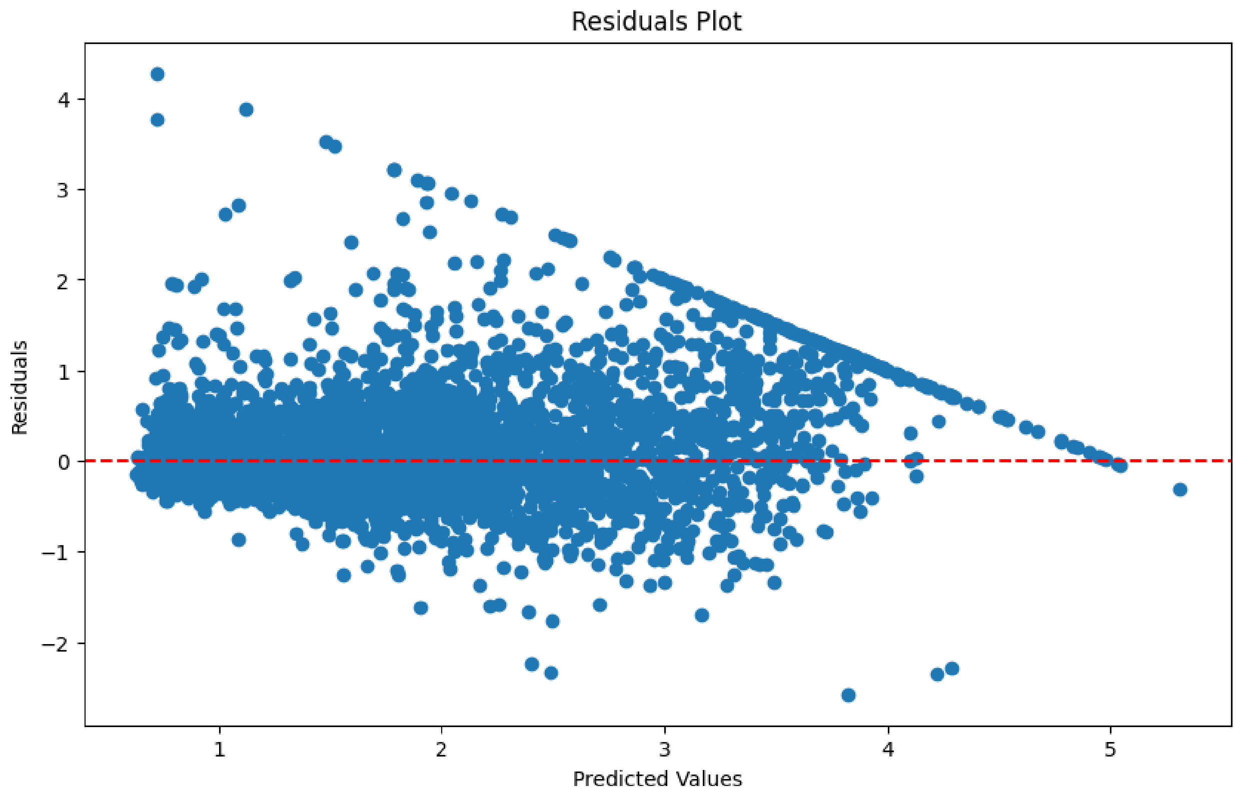
```
- val_loss: 0.3684 - val_mae: 0.4128 - learning_rate: 5.7648e-05
Epoch 19/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.4055 - mae: 0.4578
- val_loss: 0.3749 - val_mae: 0.4165 - learning_rate: 4.0354e-05
Epoch 20/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.3911 - mae: 0.4532
- val_loss: 0.3685 - val_mae: 0.4130 - learning_rate: 2.8248e-05
Epoch 21/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.4070 - mae: 0.4582
- val_loss: 0.3652 - val_mae: 0.4105 - learning_rate: 1.9773e-05
Epoch 22/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.3971 - mae: 0.4567
- val_loss: 0.3708 - val_mae: 0.4140 - learning_rate: 1.3841e-05
Epoch 23/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 3s 5ms/step - loss: 0.3955 - mae: 0.4535
- val_loss: 0.3752 - val_mae: 0.4174 - learning_rate: 9.6889e-06
Epoch 24/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.3980 - mae: 0.4530
- val_loss: 0.3657 - val_mae: 0.4106 - learning_rate: 6.7822e-06
Epoch 25/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.4011 - mae: 0.4591
- val_loss: 0.3643 - val_mae: 0.4101 - learning_rate: 4.7476e-06
Epoch 26/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.4041 - mae: 0.4604
- val_loss: 0.3777 - val_mae: 0.4193 - learning_rate: 3.3233e-06
Epoch 27/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.4008 - mae: 0.4537
- val_loss: 0.3748 - val_mae: 0.4184 - learning_rate: 2.3263e-06
Epoch 28/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.4101 - mae: 0.4598
- val_loss: 0.3752 - val_mae: 0.4182 - learning_rate: 1.6284e-06
Epoch 29/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.4174 - mae: 0.4614
- val_loss: 0.3696 - val_mae: 0.4143 - learning_rate: 1.1399e-06
Epoch 30/30
516/516 ━━━━━━━━━━━━━━━━━━━━ 3s 5ms/step - loss: 0.4060 - mae: 0.4593
- val_loss: 0.3739 - val_mae: 0.4179 - learning_rate: 7.9792e-07
129/129 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Root Mean Squared Error: 0.6115
R2 Score: 0.7147
```

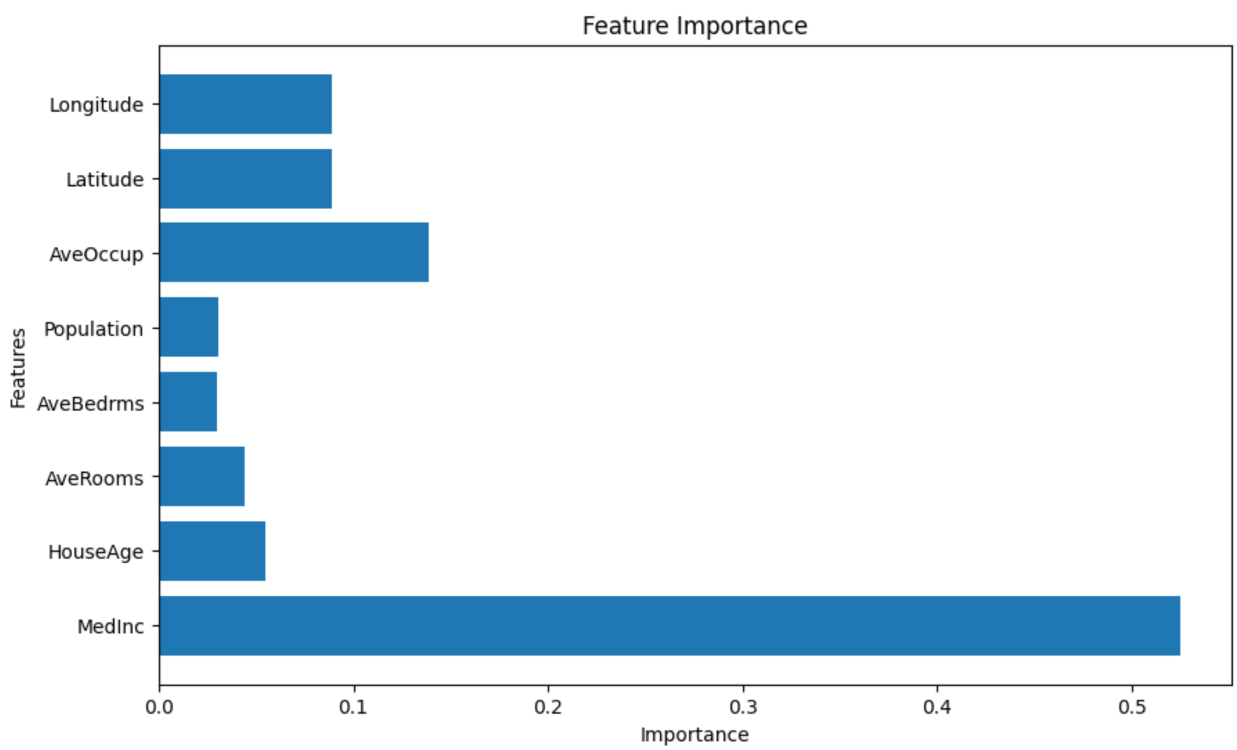

Training and Validation Loss Over Epochs
Training Loss
Validation Loss
Loss
Epochs
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30

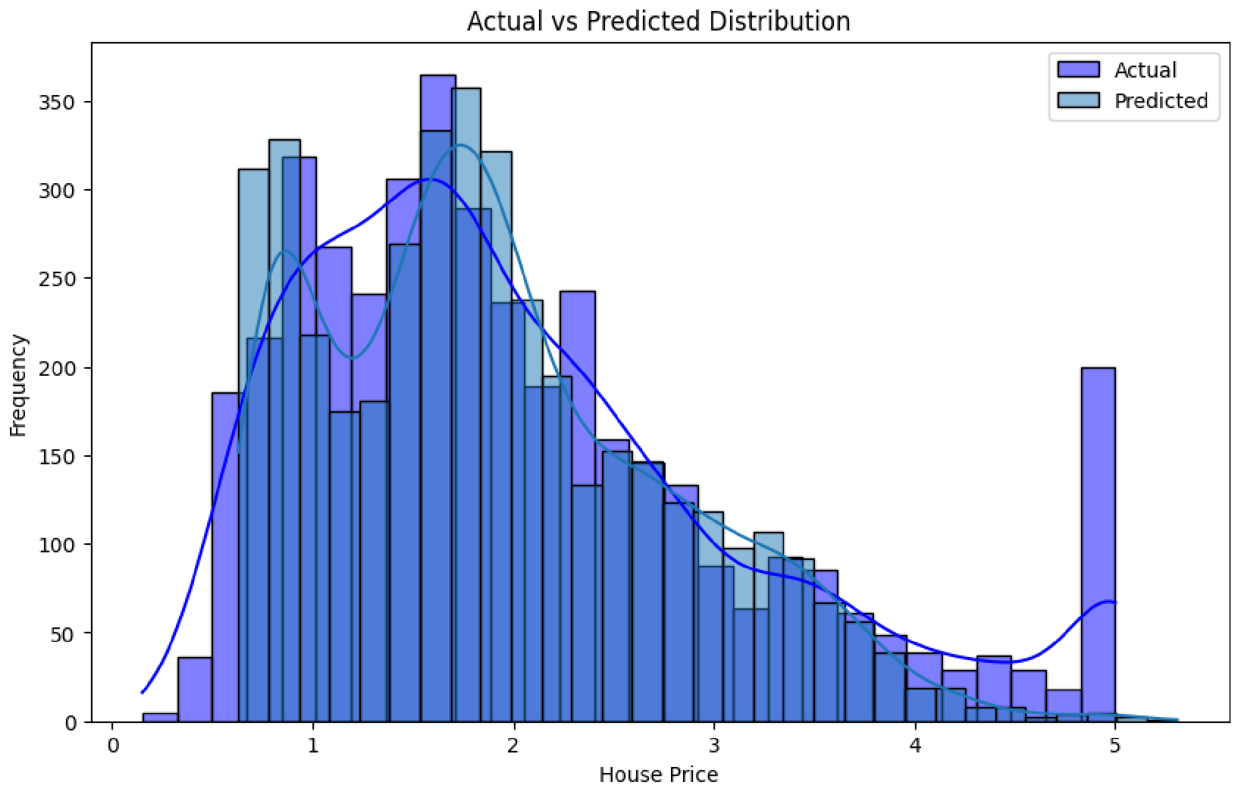

Training and Validation MAE Over Epochs
Training MAE
Validation MAE
Mean Absolute Error
Epochs
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
5
10
15
20
25
30


Learning Rate Decay Over Epochs
Learning Rate Over Epochs
Learning Rate
Epochs
0.00100
0.00095
0.00090
0.00085
0.00080
0.00075
0.00070
0
5
10
15
20
25
30


Predicted vs Actual Values
Predicted Values
Actual Values
0
1
2
3
4
5


Residuals Plot
Residuals
Predicted Values
4
3
2
1
0
−1
−2
1
2
3
4
5


Feature Importance
Features
Importance
Longitude
Latitude
AveOccup
Population
AveBedrms
AveRooms
HouseAge
MedInc
0.0
0.1
0.2
0.3
0.4
0.5



Actual vs Predicted Distribution
Actual
Predicted
Frequency
House Price
0
50
100
150
200
250
300
350
0
1
2
3
4
5